Successive Approximation in Parallel Graph Algorithms

(Extended Abstract)

Donald Fussell Ramakrishna Thurimella

Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712.

Abstract The notion of successive approximation is introduced in the context of parallel graph algorithms. The implementation of graph algorithms on Leighton's mesh of trees network model is considered. The implementations that have appeared so far in the literature are relatively straightforward. A common characteristic of these algorithms is that, in each iteration, for each vertex v, at most one edge is selected from the edges incident on v. This selection is based purely on local information such as the weights of the edges incident v or the labels of the neighboring vertices of v etc. As this sort of information appears on the same row of a mesh, these algorithms lend themselves to a direct implementation. In this paper we present an implementation of the open ear decomposition algorithm of Maon, Schieber and Vishkin. Some applications of open ear decomposition include parallel planarity testing, triconnectivity and 4-connectivity testing. This algorithm is different from the other algorithms considered for implementation on a mesh of trees in that a direct implementation is ruled out due to the communication problems posed by the network. Our implementation uses a technique of successive approximation. The process starts by finding an open ear decomposition of a subgraph of at most 2n edges: the edges of two edge-disjoint forests of G. Each subsequent iteration uses the decomposition from the previous step to obtain an open ear decomposition of an enlarged subgraph. This enlarged subgraph consists of the edges that received an ear label in the previous step together with at least as many new ones. Therefore the process converges in O(log n) iterations. The decomposition algorithm for each iteration can be distributed on the network. The whole algorithm takes O(log^3 n) time using O(n/\log n \times n/\log n) processors. Assuming adjacency matrix representation of the graph, the achieved speedup is O(log n) factor off the optimal, which is the best known.

1 Introduction

Parallel algorithms for graph problems have been the subject of much interest in recent years. Since graph algorithms are of such widespread utility as building blocks for other algorithms, it is of particular interest to determine efficient ways to perform these computations on universal parallel machines i.e. those machines which can simulate any other parallel machine built using the same hardware resources within a polylog time factor. The most commonly studied universal machine is the Parallel RAM (PRAM) shared memory model, which is the most powerful parallel machine. In recent years algorithms for finding minimum spanning forests, connected components and biconnected components in O(log n) time with O(n + e) processors have been found that run on CREW (Concurrent Read Exclusive Write) PRAM [12],[13].

While PRAMs are useful theoretical models for studying parallel algorithms, they are of limited interest in practice due to the difficulty of constructing shared memory machines with very large numbers of processors. While not as powerful as shared memory machines, network models are of interest in practical terms. A network model is a restrictive parallel computation model which

---

1This work was supported in part by the ONR under Contracts N00014-86-K-0597 and N00014-86-K-0763.
explicitly models communication and granularity constraints. The communication constraint is
due to the bounded degree of the vertices of the network. That is each processor in the network
can communicate to only a constant number of other processors in unit time. The granularity
constraint is caused by the conflicts that arise when several processors want to access the variables
that are stored in the same local memory. Such machines include Leighton’s mesh of trees and
tree of meshes [6] among others. See Fig. 1.

In this paper, the design of graph algorithms on the mesh of trees model is considered. We argue
that the idea of an approximate solution arises naturally while implementing graph algorithms on
networks and suggest, therefore, a method of successive approximation as an implementation
technique in this context.

The following algorithms have been considered for implementation on a mesh of trees. Leighton
[7] has shown that the algorithm of Hirschberg and Volper [2] for finding connected components
and minimum spanning forest of an n vertex graph can be implemented on a mesh of trees of size
n x n to run in $O(\log^2 n)$ time. Huang [3] later improved the processor-bound to $O(n^2/\log^2 n)$.
Additionally, he gives implementations on a mesh of trees that have the same time and processor
complexity for: a) the biconnected component algorithm of Tarjan and Vishkin; b) an algorithm
that finds a directed spanning forest. All these algorithms, with the exception of the biconnected
component algorithm, have the following property in common: in each iteration, for each vertex v,
an edge incident on v is picked based purely on local information. Implementation of an iteration
of these algorithms typically involves, a row of processors performing together a simple operation
on the values stored in the local memory of the processors on that row. In the case of the minimum
spanning tree algorithm, for example, each vertex v picks an edge incident on it that has the least
weight. In an implementation that means finding the minimum of at most n values that are stored
on same row of the mesh. These operations, usually, take no more than $\log n$ steps. In the case of
the biconnected component algorithm, Tarjan and Vishkin make an important observation about
the blocks of a graph which implies the property mentioned above. See the construction of $G''$ for
details [13].

An open ear decomposition of an undirected graph, roughly speaking, is a partition of $E(G)$
into an ordered collection of edge disjoint simple paths that satisfy certain requirements. Ear
decomposition was defined by Whitney in [14] in 1932. Recently, Lovasz showed [8] that the prob-
lem of ear decomposition has a parallel algorithm which runs in poly-log time using a polynomial
number of processors and is therefore in the class NC. Later Maon, Schieber and Vishkin (MSV)
[9], and Miller and Ramachandran [10] gave algorithms for finding an open ear decomposition
which run in $O(\log n)$ time using $n + m$ processors on a CRCW PRAM. We consider, in this
paper, an implementation of MSV’s algorithm. There are two important reasons for this choice:
first, it does not satisfy the property mentioned in the previous paragraph and therefore poses
challenging implementation problems; second, the idea of partitioning the edges into open ears
has been found to be extremely useful in the design of parallel graph algorithms.

Maon et al. conjecture that, Depth First Search (DFS) being “inherently sequential”, open ear
decomposition may prove to be as useful in the design of parallel graph algorithms as DFS in the
design of sequential graph algorithms. In fact, parallel graph algorithms that are based on open
ear decomposition include algorithms for testing a graph for planarity [5], triconnectivity [1], [11]
and 4-connectivity [4].

A graph has an open ear decomposition iff it is biconnected [14]. An ear decomposition of
a connected, bridgeless graph $G$ is considered to be open if the number of closed ears in that
decomposition is equal to the number of blocks of that graph. Therefore an ear decomposition in
which the number of closed ears is greater than the number of blocks of $G$ is, in a sense, only an
approximation to an open ear decomposition. In other words, between two ear decompositions of
a graph, the one with the least number of closed ears is better. Our algorithm starts by finding an
open ear decomposition of a subgraph $G'$ that is the union of a spanning tree $T$ and a spanning